Teemu Ikonen

SETUP-TIETOKANTA JA KÄYTTÖLIITTYMÄ

Insinöörityö
Kajaanin ammattikorkeakoulu
Tekniikka ja liikenne
Ajoneuvojen tietojärjestelmät
2013

KAJAANIN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ
TIIVISTELMÄ

Koulutusala Tekniikka ja liikenne	Koulutusohjelma Tietotekniikan koulutusohjelma
Tekijä(t) Teemu Ikonen	
Työn nimi Setup-tietokanta ja käyttöliittymä	
Vaihtoehtoiset ammattiopinnot Ajoneuvojen tietojärjestelmät	Ohjaaja(t) Arto Partanen
	Toimeksiantaja PrintSport Oy / Eero Räikkönen
Aika 15.11.2013	Sivumäärä ja liitteet 35

Tämän insinöörityön tavoitteena oli suunnitella ja toteuttaa tietokanta moottoriurheilussa käytettävien ajoneuvojen säätöarvoille. Tietokannan lisäksi tehtävänä oli ohjelmoida käyttämisen mahdollistava käyttöliittymä. Toimeksiantajana toimi Lievestuoreella sijaitseva PrintSport Oy, joka on moottoriurheiluun erikoistunut yritys.

Ennen käytännön työn aloitusta kartoitettiin erilaisia mahdollisuuksia työn toteuttamiseksi. Työkaluissa päädyttiin käyttämään avoimen lähdekoodin MySQL Workbenchiä tietokantojen hallintajärjestelmänä, ja käyttöliittymän tekoon Qt-kehitysympäristöä.

Insinöörityössä tutustutaan relaatiotietokantojen teoriaan, yleisesti käytettyihin käsitteisiin ja tietokantojen suunnittelussa huomioon otettaviin perusasioihin. Käyttöliittymä suunniteltiin käytettävyyden ja helppokäyttöisyyden näkökulmasta, joita myös sivutaan tässä työssä. Insinöörityön aikana suunniteltiin ja toteutettiin paikallisesti toimiva MySQL-tietokanta ja Qt-kehitysympäristössä käyttöliittymä jotka on dokumentoitu tähän teokseen.

Lopputuloksena toteutettiin toimiva relaatiotietokanta ja sen käytön mahdollistava käyttöliittymä. Käyttöliittymän kautta saadaan tallennettua uusia setupeja tietokantaan ja selattua sekä poistettua aiemmin tallennettuja.

Kieli	Suomi
Asiasanat	relaatiotietokanta, MySQL, käyttöliittymä, ohjelmointi, Qt
Säilytyspaikka	☒ Verkkokirjasto Theseus ☐ Kajaanin ammattikorkeakoulun kirjasto

KAJAANIN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

THESIS
ABSTRACT

School Engineering	Degree Programme Information Technology
Author(s) Teemu Ikonen	
Title Setup Database and User Interface	
Optional Professional Studies Vehicle Information Systems	Instructor(s) Mr Arto Partanen
	Commissioned by PrintSport Oy / Mr Eero Räikkönen
Date 15 November 2013	Total Number of Pages and Appendices 35

The aim of this Bachelor's thesis was to design and implement a database for setup values used in motorsport vehicles. In addition to the database, the task was to program a user interface to enable the use of the database. This thesis was commissioned by PrintSport Oy, a company located in Lievestuore and specialized in motorsports.

Before starting the practical part of the thesis different possibilities had to be studied for carrying out the work. The survey resulted in the use of open source MySQL Workbench database management system, and the Qt development environment for designing the user interface.

The thesis introduces the theory of relational databases, general terms used in the database and the basics to be considered during planning. The user interface was designed from the usability and ease of use points of view also dealt with in this document. During the making of the thesis a locally-based MySQL database and a user interface with Qt development environment were planned and accomplished.

As a result, a working relational database was implemented and a user interface to access the database. The user interface is used to save new setups in the database, browse through previously saved ones or to delete them.

Language of Thesis	Finnish
Keywords	relational database, MySQL, user interface, programming, Qt
Deposited at	☒ Electronic library Theseus ☐ Library of Kajaani University of Applied Sciences

ALKUSANAT

Tämän insinöörityön teko on ollut suurin yksittäinen projekti, jonka olen saanut toteuttaa suurimmalta osalta itsenäisesti. Työn aiheesta ja sen teon mahdollistamisesta haluan kiittää PrintSport Oy:n Eero Räikköstä. Suuri kiitos kuuluu Qt:lla ohjelmoinnissa opastaneelle, tulevalle ohjelmointigurulle Tinja Paavosepälle, sekä työn ohjaajana toimineelle Arto Partaselle. Lisäksi haluan kiittää kaikkia ystäviäni, jotka ovat luottaneet ja uskoneet silloinkin, kun itse on epäröinyt.

Kajaanissa, 21.11.2013

Teemu Ikonen

SISÄLLYS

SYMBOLILUETTELO

Avoin lähdekoodi	(Engl. open source) Sellaisella tuottamis- ja kehitysmenetelmällä tehty ohjelma, jonka lisenssi antaa vapauden käyttää, kopioida, levittää ja muokata sitä
C++	Yleiskäyttöinen, alustariippumaton suosittu ohjelmointikieli
Dialogi	Ohjelmaikkuna, jonka kautta käyttäjä ja ohjelma ovat vuorovaikutuksessa
ER-kaavio	(Engl. entity-relationship model) Kaaviotyyppi, jota käytetään muun muassa tietokannan mallinnuksessa
Kehitysympäristö	Ohjelmistojen suunnitteluun ja toteutukseen tarkoitettu ohjelma
Käyttöliittymä	Ohjelman osa, joka näkyy käyttäjälle ja jonka kautta ohjelmaa käytetään
Luokka	Olion tyyppi olio-ohjelmoinnissa
Metodi	Luokasta löytyvä funktio, aliohjelma
MySQL	Avoimen lähdekoodin relaatiotietokantojen hallintaohjelmisto
Qt	(Q toolkit tai ”cute”), ohjelmien kehitysympäristö
Setup	(Suom. asetukset) Kokoelma säätöarvoja, joilla optimoidaan muun muassa suorituskykyä ja hallittavuutta
Setup sheet	Lomake, johon setupissa käytetyt säätöarvot kerätään
SQL	(Structured Query Language) Standardoitu tietokannan kyselykieli
Tietokanta	Tietovarasto
Widget	Graafisen käyttöliittymän yksittäinen elementti, jolla käyttäjä vaikuttaa ohjelman suoritukseen

1 JOHDANTO

Tämän insinöörityön tarkoituksena oli toteuttaa PrintSport Oy:n mekaanikoiden käyttöön tietokanta, johon voidaan tallentaa jälkikäteen rallin aikana autoissa käytetyt säätöarvot eli setupit. Ne lisätään tietokantaan yksinkertaisen ja helppokäyttöisen käyttöliittymän kautta, jonka suunnittelu ja toteutus ovat osa tätä insinöörityötä.

PrintSport Oy on Lievestuoreella sijaitseva vuonna 1997 perustettu yritys, joka on erikoistunut moottoriurheiluun. Yritys myy asiakkailleen huoltopalveluja omissa tiloissaan, sekä keikkaluontoisesti testeissä tai yksittäisissä kilpailuissa ja ralleissa. Tämän lisäksi PrintSport Oy myy varaosia huoltamiinsa automerkkeihin ja toimii myös useiden myyntiartikkeleiden maahantuojana. Laajan autourheilun tekniikan hallinnan lisäksi yritys tarjoaa palveluna myös muun muassa ajokoulutusta.

Tietojen keräämisen ja tallettamisen tarkoituksena on toimia muistin tukena, jolloin voidaan palata selaamaan aiemmin käytettyjä setupeja ja siten käyttää niitä apuna valmistauduttaessa uuteen ralliin. Setupit ovat kokoelmia ralliautoissa käytetyistä säätöarvoista, joita käytetään säädettäessä muun muassa auton alustaa, ohjausta ja vaihteistoa. Autot ovat pääasiassa mekaanikkokohtaisia, jolloin auton oman mekaanikon ollessa estynyt voi toinen mekaanikko käydä tutustumassa autossa käytettyihin setupeihin. Yrityksellä ei ole aiemmin ollut käytössä yhteisesti sovittua tapaa tallentaa setupeja, vaan tietoja on pidetty joko oman muistin tai muistilappujen varassa.

Insinöörityön alkupuolella kerrotaan teoriaa relaatiotietokannoista ja Qt-kehitysympäristöstä. Näiden teoriassa opeteltujen menetelmien avustuksella lähdettiin suunnittelemaan ja toteuttamaan lopputyön varsinaista käytännön osuutta.

2 RELAATIOTIETOKANTA

Kun tehtävän luonne oli tiedossa ja tiedettiin, millaisen tuotteen PrintSport Oy (jatkossa asiakas) tarvitsee, voitiin alkaa miettiä ja rajata välineitä, joilla tällainen tehtävä on mahdollista toteuttaa. Alkutietojen perusteella sovelluksella on tarkoitus tallettaa ja käsitellä tietoja, joten loogisena lisänä käyttöliittymälle tarvitaan myös jokin tapa varastoida tietoja, tietokanta. Asiakas lähtikin jo aikaisessa vaiheessa ehdottamaan tietokannan toteutusta, eikä muita varteenotettavia vaihtoehtoja käynyt mielessäkään. Tietokanta muodostuu tallennetusta joukosta tietoa, joilla on looginen yhteys toisiinsa. Tietojoukkoa voidaan käsitellä tietokantakielellä (SQL) ja hallinnoida tietokannan hallintajärjestelmällä. Asiakkaalta saadusta materiaalista huomasi jo aikaisessa vaiheessa, että käsiteltävää toisiinsa liittyvää tietoa on paljon ja ajan saatossa sitä tulee kertymään paljon lisää, joten tämä oli otettava huomioon suunnittelussa. Relaatiotietokanta oli siten järkevä valinta helppokäyttöisyyden ja suorituskykynsä vuoksi. [1, s. 4.]

Relaatiotietokanta on ollut olemassa yli 25 vuotta. Se on muodostanut ympärilleen miljardiluokan teollisuuden ja on nykyisin laajin käytetty tietokantatyyppi maailmassa ja samalla oleellinen osa jokapäiväistä elämää. Relaatiotietokantaa käytetään esimerkiksi ostettaessa tavaraa kaupasta, tehtäessä lomasuunnitelmia matkanjärjestäjän kanssa tai käytettäessä verkkokaupan palveluita. [2, s. 3.]

2.1 Relaatiomalli

Vuonna 1969 IBM:n tutkija, tohtori Edgar F. Codd, synnytti relaatiomallin, johon perustuu ja jolla määritellään relaatiotietokantojen teoreettinen pohja. Tohtori Codd toimi IBM:n kehitysyksikössä 60-luvun lopulla ja etsi uusia tapoja käsitellä suuria määriä tietoa. Matemaatikkona hän uskoi vahvasti voivansa soveltaa tiettyjä matematiikan haaroja ratkaistakseen ongelmia, joita aiheuttivat muun muassa tiedon toistuvuus ja heikko yhteiskäyttö sekä tietokannan rakenteen yliriippuvuus sen fyysiseen toteutukseen. Codd perusti uuden mallinsa kahteen matematiikan haaraan: joukko-oppiin ja predikaattilogiikkaan. Mallin nimi johtaakin joukko-opin termiin relaatio eikä virheellisesti luultuun suhteeseen (engl. relation), joka relaatiotietokannan tauluilla on toisiinsa. [2, s. 12-13.]

Relaatiotietokanta tallettaa tietoa relaatioihin, jotka käyttäjä näkee tauluina. Jokainen relaatio koostuu riveistä ja sarakkeista. Rivien tai sarakkeiden fyysinen järjestys taulussa on täysin merkityksetön, ja jokainen rivi tunnistetaan perusavaimella (primary key, PK), joka sisältää uniikin arvon. Nämä kaksi erityispiirrettä mahdollistavat relaatiotietokannan sisältämän tiedon itsenäisen olemassaolon riippumatta siitä, kuinka ne on fyysisesti varastoitu tietokoneelle. [2, s. 13.]

2.2 Rakenne

Relaatiotietokantojen suunnitteluun liittyviin käsitteisiin on olemassa tiettyjä termejä, kuten missä tahansa ammattisanastossa. Seuraavissa kappaleissa pyritään avaamaan tietokannan taulujen rakenteita kuvaavia termejä ja niiden tarkoitusta tietokannassa.

Taulu

Taulut ovat tietokannan tärkein rakenne, ja jokainen taulu edustaa aina yksittäistä, tiettyä aihetta. Jokainen taulu sisältää aina vähintään yhden sarakkeen, jota kutsutaan perusavaimeksi (kuvassa 1 perusavain on asiakas_id). Taulun edustama aihe voi olla joko olio tai tapahtuma. Kun aiheena on olio, taulu edustaa jotain konkreettista, kuten henkilöä, paikkaa tai esinettä. Olioon liittyvät ominaisuudet voidaan tallentaa tiedoksi, ja tätä tietoa voidaan sen jälkeen käsitellä lukuisilla eri tavoilla. Kuvassa 1 on esimerkki yleisestä oliotyyppisestä taulusta. [2, s. 53.]

asiakas — perusavainsarake

asiakas_id	enimi	snimi	kunta
5001	Jukka	Korpi	Sotkamo
5002	Pekka	Mäki	Kuhmo
5003	Pentti	Suopunki	Oulu
5004	Erkki	Maasara	Kajaani

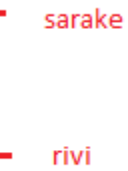
Kuva 1. Yleinen oliotyyppinen taulu

Kun aiheena on tapahtuma, taulu edustaa tiettynä ajankohtana tapahtuvaa asiaa, jonka yksityiskohdat voidaan tallentaa. Yksityiskohdat voidaan tallentaa tiedoksi ja käsitellä samaan tyyliin kuin oliotyyppisen taulun sisältö. Esimerkkeinä tallennettavista tapahtumista mainitta-

koon juridinen kuulustelu, rahoituksen jakaminen, laboratoriotestin tulokset ja geologinen kartoitus. Kuvassa 2 on taulu, joka havainnollistaa tapahtumaa. [2, s. 53.]

potilas_tapaaminen

potilas_id	pvm	aika	tohtori	verenpaine
24002	25.2.2001	10:30	Sivonen	120/80
24394	1.3.2001	15:15	Kaikkonen	112/75
24667	1.3.2001	15:50	Kaikkonen	130/82
24112	5.3.2001	8:45	Kemppainen	110/75



Kuva 2. Tapahtumatyyppinen taulu

Sarake

Sarake on pienin rakenne tietokannassa, ja sillä erotellaan taulun aiheen ominaisarvot. Sarakkeet ovat rakenteita, joihin tieto todella talletetaan. Sarakkeiden sisältämät tiedot voidaan noutaa ja esittää informaationa. Tiedosta saadun informaation laadulla on suora yhteys aikaan, joka on käytetty sarakkeen rakenteen ja tiedon eheyden varmistamiseen.

Jokainen sarake huolellisesti suunnitellussa tietokannassa sisältää ainoastaan yhden arvon ja sarakkeen nimi kertoo, minkä tyyppinen arvo on. Kuvaavasti nimettyyn sarakkeeseen on helppo syöttää tietoa, mutta myös lajitella ja etsiä sitä. [2, s. 55.]

Rivi

Rivi esittää uniikin instanssin taulun aiheesta. Se koostuu taulun kaikista sarakkeista riippumatta siitä, sisältääkö sarake arvoja vai ei. Taulun määrittelytavasta johtuen jokainen rivi tunnistetaan uniikin perusavaimen arvon perusteella läpi koko tietokannan. Rivit ovat tärkeä tekijä taulujen suhteiden ymmärtämisessä, koska on tarve tietää, kuinka kahden eri taulun rivit liittyvät toisiinsa. [2, s. 56-57.]

Näkymä

Näkymä on ”virtuaalinen taulu”, johon koostetaan sarakkeita yhdestä tai useammasta tietokannan taulusta. Relaatiomallissa näkymää sanotaan virtuaaliseksi, koska se noutaa tietoa varsinaisista tauluista sen sijaan, että tallettaisi sitä itse. Ainoa tieto, joka näkymästä tietokantaan

tallentuu, on sen rakenne. Näkymän avulla voidaan nähdä tietokannan tietoa useista eri näkökulmista, ja se tarjoaakin joustavuutta tiedon kanssa työskentelyyn. Näkymien luontiin on monta tapaa, ja ne ovat erityisen hyödyllisiä pohjautuessaan useampaan toisiinsa liittyvään tauluun. [2, s. 57.]

Avaimet

Avaimet ovat erityisiä sarakkeita, jotka ovat tärkeässä roolissa ja avaimen tyyppi määrittää sen tarkoituksen taulussa. On olemassa muutamia avaintyyppejä, mutta kaksi kaikkein tärkeintä ovat perusavain ja viiteavain.

Perusavain on sarake tai ryhmä sarakkeita, jolla yksilöivästi tunnistetaan jokainen rivi taulussa. Perusavainsarakkeessa ei siis saa olla kahdella tai useammalla rivillä samaa arvoa. Perusavain on ehdottomasti tärkein avain, ja sen tulisikin löytyä tietokannan jokaisesta taulusta.

Kun päätetään kahden taulun välisestä suhteesta toisiinsa, yhteys muodostetaan yleensä kopioimalla ensimmäisen taulun perusavain ja liitetään se toisen taulun rakenteeseen, jolloin siitä tulee (toisen taulun) viiteavain. [2, s. 59-60.]

2.3 Yhteydet

Kahden taulun välillä on yhteys silloin, kun ensimmäisen taulun rivejä voidaan liittää toisesta taulusta löytyviin riveihin. Yhteys voidaan muodostaa perus- ja viiteavaimilla tai erillisen kolmannen taulun kautta, jota kutsutaan assosiatiiviseksi tauluksi, välitauluksi. Tapa, jolla taulujen välinen yhteys muodostetaan riippuu taulujen yhteyden tyypistä. Taulujen välisiä yhteystyyppejä on kolme: yksi-yhteen, yksi-moneen ja moni-moneen. [2, s. 62-63.]

Yksi-yhteen-yhteys

Taulujen välillä on yksi-yhteen-yhteys, kun ensimmäisen taulun yksi rivi liittyy ainoastaan yhteen riviin toisessa taulussa ja päinvastoin. Tämän tyyppisessä yhteydessä (isä-lapsi-yhteys) yksi taulu toimii niin kutsutusti isätauluna ja toinen lapsitauluna. Yhteys muodostetaan ottamalla kopio isätaulun perusavaimesta ja sisällyttämällä se lapsitaulun rakenteeseen viiteavaimeksi. Tässä yhteystyypissä kumpikin taulu voi jakaa saman perusavaimen. [2, s. 63.]

Yksi-moneen-yhteys

Taulujen välillä on yksi-moneen-yhteys, kun ensimmäisen taulun yksi rivi voi liittyä useisiin riveihin toisessa taulussa, mutta yksi rivi toisessa taulussa voi liittyä ainoastaan yhteen riviin ensimmäisessä taulussa. Kuten yksi-yhteen-yhteys, myös tämä voidaan ajatella isä-lapsi-yhteytenä, jolloin yhdellä isätaululla voi olla monta lapsitaulua, mutta yhdellä lapsitaululla voi olla ainoastaan yksi isätaulu. Yksi-moneen-yhteys muodostetaan kopioimalla isätaulun perusavain ja sisällyttämällä se lapsitauluun viiteavaimeksi. Tämä on yleisin taulujen välinen yhteystyyppi, jota tietokannassa käytetään, koska se auttaa pitämään ylimääräisen tiedon määrän mahdollisimman vähäisenä. [2, s. 64-65.]

Moni-moneen-yhteys

Taulujen välillä on moni-moneen-yhteys, kun ensimmäisen taulun rivi voi liittyä useampaan riviin toisessa taulussa ja päinvastoin. Moni-moneen-yhteys muodostetaan välitaulun kautta. Välitaulu helpottaa kahden taulun rivien yhteenliittämisessä ja varmistaa, ettei tietojen lisäyksessä, poistamisessa tai muokkaamisessa tule ongelmia. Välitaulu määritellään ottamalla kopiot kummankin taulun perusavaimesta ja lisäämällä ne muodostuvan uuden taulun rakenteeseen. Näillä sarakkeilla on kaksi erillistä roolia: yhdessä ne muodostavat moniosaisen perusavaimen ja ollessaan erillään kumpikin toimii viiteavaimena. [2, s. 65-66.]

2.4 Suunnittelun vaiheet

Tietokannan suunnitteluprosessin alussa määritellään tietokannan johtoajatus ja tehtävän tavoitteet. Jokainen tietokanta luodaan tiettyä tarkoitusta varten. Johtoajatuksella tunnistetaan ja vahvistetaan tietokannan tarkoitus, ja se tarjoaa suunnittelutyölle selvän painopisteen. Tämä auttaa varmistamaan, että prosessissa kehitetään tehtävään sopiva tietokantarakenne ja kerättävät tiedot ovat tarkoituksen mukaiset. Tietokannan tehtävän tavoitteilla tarkoitetaan niitä tehtäviä, joita käyttäjät suorittavat tietokannan tiedoilla. Nämä tehtävät tukevat johtoajatusta ja helpottavat määrittämään tietokannan rakennetta. [2, s. 79-80.]

Johtoajatuksen ja tavoitteiden selkiydyttyä tutustutaan tämänhetkiseen tietokantaan, jos sellaista on. Organisaatiosta riippuen se on tyypillisesti vanhalla tekniikalla toteutettu tietokanta tai paperiarkisto. Käytössä olevan menetelmän analysointi tuottaa arvokasta tietoa siitä,

kuinka organisaatio käyttää ja hallinnoi tietoja. Analyysillä selviää myös tietojen keräyksen ja esittämisen nykytila, jota voidaan selvittää järjestämällä haastatteluita käyttäjien ja johdon kanssa. Kysytyillä kysymyksillä saaduilla tiedoilla on suuri vaikutus tietokannan lopulliseen rakenteeseen, kuten myös kysymättä jääneillä kysymyksillä ja saamatta jääneillä tiedoilla. [2, s. 80-81.]

Kun suunnittelun alkuun tarvittavat perustiedot ja rajaukset alkavat olla selvillä, voidaan aloittaa niin kutsuttu suunnitteluputki. Se sisältää viisi eri vaihetta, jotka joiltain osin sisältyvät kaikkiin tietokantojen suunnitteluhankkeisiin. Yksi vaihe johtaa yleensä aina tilaan tai malliin, josta seuraavan vaiheen aloitus mahdollistuu. [1, s. 24.]

Suunnitteluputken ensimmäiset kolme vaihetta keskittyvät tuoteriippumattomaan loogiseen suunnitteluun, joka alkaa käsiteanalyysin teolla. Käsiteanalyysin tuloksena muodostetaan kohdealueen käsitteiden mukaan karkean tason käsitemalli, jossa suunnitellaan muun muassa taulujen välisiä yhteyksiä. Käsitemallia tarkennetaan tietotarveanalyysillä, jossa lisätään vielä puuttuvia tietoja ja yhteyksiä, joihin on ilmennyt tarve. Täydennetty käsitemalli on valmis, kun kaikki puuttuvat tiedot on lisätty. Tämän jälkeen voidaan aloittaa normalisointi, jolla pyritään poistamaan tietokannasta turha tietojen toisteisuus. Normalisoinnin myötä tietokannan looginen suunnittelu päättyy ja siirrytään tuotekohtaiseen fyysiseen suunnitteluun. [1, s. 24-25.]

Viimeistään fyysisen suunnittelun alkaessa pitää olla selvillä, mitä tietokannanhallintajärjestelmää käytetään tietokannan toteutukseen ja hallinnointiin. Normalisoitu käsitemalli muutetaan tiettyjen sääntöjen mukaisesti tietokannan tauluiksi (ER-kaavioksi). Fyysisessä suunnittelussa otetaan huomioon myös suorituskykyyn liittyvät asiat. [1, s. 25.]

Tietokannan suunnitteluputki ei etene vesiputousmaisesti, vaan menetelmissä palataan välillä esiin nousevien tarpeiden mukaan takaisin edellisiin vaiheisiin. Iterointikierrosten myötä tarkentuvat määrittely, suunnittelu ja toteutus. Isompia tietojärjestelmiä rakennettaessa voidaan hyödyntää inkrementaalista lähestymistapaa, jossa järjestelmä koostetaan pienemmistä kokonaisuuksista. [1, s. 25.]

2.5 SQL-kieli

SQL (Structured Query Language) on standardikieli tietokantojen käyttöön ja käsittelyyn. Kieli koostuu erilaisista kyselyistä, joilla vaikutetaan tietokannan rakenteeseen ja sisältöön. Käytetyimmät perusoperaatiot ovat kuitenkin SELECT (tiedonhaku), UPDATE (päivitys), DELETE (poisto) ja INSERT (tiedon lisäys). [3.]

Perusoperaatioita havainnollistavat esimerkit koskevat sivulta 3 löytyvän kuvan 1 mukaista taulua.

SELECT

SELECT-kyselyllä valitaan eli käytännössä haetaan tietoa tietokannasta.

SELECT enimi, snimi **FROM** asiakas;

Tällä kyselyllä haetaan sarakkeet ”enimi” ja ”snimi” taulusta ”asiakas” ja kaikki löytyneet rivit näytetään. Kyselyyn voidaan lisätä ehto jatkamalla kyselyä sanalla WHERE.

SELECT enimi, snimi **FROM** asiakas **WHERE** kunta = ’Oulu’;

Ehdon lisäyksellä saadaan muuten sama tulos kuin edellä, mutta lopputuloksena näytetään ainoastaan rivit, joilla sarakkeessa ”kunta” on arvona Oulu.

UPDATE

UPDATE-kyselyllä päivitetään tauluun aiemmin syötettyjä tietoja.

UPDATE asiakas **SET** snimi = ’Mäkinen’ **WHERE** asiakas_id = 5002;

Tämä kysely päivittää taulun ”asiakas” sarakkeen ”snimi” arvoksi Mäkinen riviltä, jossa ”asiakas_id” on arvoltaan 5002.

DELETE

DELETE-kyselyllä poistetaan tietokannan taulusta rivi tai useampia.

DELETE FROM asiakas **WHERE** asiakas_id = 5004;

Kysely poistaa “asiakas”-taulusta koko rivin, jolla “asiakas_id” arvo on 5004, eli kaikki Erkki Maasaran tiedot häviävät.

INSERT

INSERT-kyselyllä lisätään tietokannan tauluun uusi rivi.

INSERT INTO asiakas (enimi, snimi, kunta) **VALUES** (’Jorma’, ’Uotinen’, ’Pori’);

Kysely lisää rivin ”asiakas”-tauluun ja asettaa arvoiksi sarakkeisiin ”enimi” = Jorma, ”snimi” = Uotinen ja ”kunta” = Pori. Sarakkeeseen ”asiakas_id” ei lisätty arvoa, koska oletetaan, että tämän taulun perusavaimena toimiva indeksinumero on asetettu juoksevaksi. Tällöin tietokanta osaa asettaa sille annettujen sääntöjen mukaisesti uuden rivin indeksin arvoksi 5005, seuraavaksi 5006 ja niin edelleen.

3 KÄYTTÖLIITTYMÄ

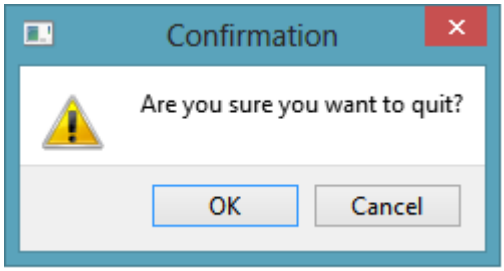
Käyttöliittymä mahdollistaa tietokoneohjelman esittämisen käyttäjälle graafisena. Se on siis ohjelman osa, joka näkyy ulospäin käyttäjälle ja jonka välityksellä ohjelmaa käytetään. Graafinen käyttöliittymä sisältää ikkunoita, valikoita, tekstikenttiä ja painikkeita - eli kaikkea, jolla käyttäjä pääsee vaikuttamaan ohjelman ajoon. Huolellisella ja monipuolisella suunnittelulla saadaan aikaan toimiva käyttöliittymä, jonka käyttö parantaa tuottavuutta. [4, s. 7.]

3.1 Käytettävyys

Tärkeimpänä vaatimuksena tämän käyttöliittymän suunnittelussa on lopputuotteen helppokäyttöisyys. Käyttöliittymän kautta syötetään tallennettavat tiedot tietokantaan tai selataan ja päivitetään aiemmin syötettyjä tietoja. Tiedot näkyvät yksittäin auto- ja tapahtumakohtaisesti. Koska yhden kisapäivän ja tapahtuman aikana saatetaan käyttää useampia eri setupeja, erotellaan nämä toisistaan kellonajan perusteella.

Käyttöliittymän ulkonäköä ja interaktiivisuutta suunniteltaessa helppokäyttöisyyden näkökulmasta tulee ensimmäisenä mieleen yksinkertaisuus. Mitä yksinkertaisempana käyttäjälle näkyvän ohjelman saa pidettyä, sitä helpompaa on myös sen käyttäminen. Kun tämän ajattelumallin lisäksi otetaan huomioon aiemmat tietotekniset käyttökokemukset, saadaan toivottavasti käytettävyyden kannalta onnistunut lopputulos. Aiemmilla käyttökokemuksilla tarkoitetaan sitä, mihin käyttäjät ovat tottuneet työskennellessään tietokoneohjelmien parissa. Tottumusten myötä ihmisen ajattelulle kehittyy tietynlaiset ennakko-odotukset, esimerkiksi seuraava: ohjelma suljetaan painamalla ohjelmaikkunan oikeassa yläkulmassa sijaitsevaa punaisella pohjalla olevaa vinoristiä.

Valintapainikkeiden sijoittelu ja looginen mutta myös tottumukseen pohjautuva järjestely on tärkeää ottaa huomioon. Tämä helpottaa ohjelman käyttämisen oppimista ja vähentää muutosvastaisuutta, johon usein törmää varsinkin työskenneltäessä tietoteknisten välineiden kanssa. Painikkeiden nimeäminen yleisesti käytetyillä, toimintoa kuvaavilla sanoilla on myös tehokas tapa helpottaa uuden sovelluksen käyttöönottoa. Esimerkkinä kuvassa 3 on valintaikkuna, jossa OK- ja Cancel-painikkeet on sijoitettu Windows-käyttöympäristössä tutun standardin mukaisesti.



Kuva 3. Yleisesti käytetty valintaikkuna

Käyttöliittymän tapauksessa käytettävyydellä kuvataan laatua eli sitä, kuinka hyvin käyttäjän tarpeet saadaan sovelluksessa täytettyä. Käytettävyyttä itsessään ei saada mitattua, mutta on olemassa käytettävyysparametreja, joita mittaamalla saadaan käytettävyyden laatua arvioitua. Parametrit liittyvät käyttäjäkokemukseen subjektiivisesti tyytyväisyyden kannalta ja objektiivisesti työn tehokkuutta sekä tehtävistä suoriutumista seuraamalla. Sovelluksia vertailtaessa ei välttämättä kiinnitetä huomiota pelkästään toiminnallisuuteen, vaan käytettävyys on isossa osassa. [4, s. 8.]

3.2 Käytettävyystekijät

Nielsenin määrittämät yleiset käytettävyystekijät on jaettu viiteen, jollain tavalla mitattavissa olevaan joukkoon. Nämä joukot ovat opittavuus, tehokkuus, muistettavuus, virheettömyys ja tyytyväisyys. [4, s. 22.]

Opittavuudella viitataan järjestelmän käytön oppimisen nopeuteen ja helppouteen. Tehokkuus mittaa järjestelmän toimivuutta siinä vaiheessa, kun käyttö on jo opittu, eli kuinka sujuvasti ja nopeasti käyttäjä järjestelmää käyttää. Muistettavuus tulee vastaan silloin, kun järjestelmää käytetään satunnaisesti: miten hyvin käyttö jää mieleen ja onnistuuko tehtävien hoitaminen käyttäjältä vaikkapa kuukauden tauon jälkeen. Virheettömyydellä pyritään selkeään ja johdonmukaiseen järjestelmään, jotta käyttäjän tekemien virheiden määrä saadaan pidettyä mahdollisimman pienenä. Kun käyttäjä on tyytyväinen käyttämäänsä järjestelmään, hän toimii tehokkaasti ja tekee asiat mielellään. [4, s. 22-24.]

4 QT-OHJELMOINTI

Qt on alustariippumaton, graafisen käyttöliittymän (Qt Creator) sisältävä kehitysympäristö, jota käytetään laajasti sovellusohjelmien toteutukseen. Qt:lla voidaan kehittää myös käyttöliittymättömiä ohjelmia, kuten komentorivityökaluja ja konsoleita palvelinkäyttöön. Ohjelmakehityksessä voidaan käyttää C++:n lisäksi myös useita muita eri ohjelmointikieliä (muun muassa C#, Java ja Python), ja se toimii monissa käyttöjärjestelmissä, kuten Windows, Symbian, OS X ja Linux (sulautetuille järjestelmille). Ohjelmointi on tehokasta, sillä jokaiselle käyttöjärjestelmäalustalle ei tarvitse tehdä erillistä lähdekoodia, vaan yhdestä lähteestä saadaan toimiva ohjelma kääntämällä se uudelleen käytettäväksi toisessa käyttöjärjestelmässä. [5.]

4.1 Historiaa

Qt-kehitysympäristö tuli julkisesti saataville ensimmäisen kerran toukokuussa 1995. Ympäristön isät ja alkuperäiset kehittäjät olivat norjalaiset tietotekniikan maisterit Haavard Nord ja Eirik Chambe-Eng. Idea tällaiselle järjestelmälle syntyi miesten työskennellessä C++-kielisen, käyttöliittymällisen tietokantasovelluksen parissa, joka piti saada käännettyä toimimaan useammalla alustalla. Haavard alkoi vuonna 1991 kirjoittaa luokkia, joista myöhemmin muodostui Qt. Muutamia vuosia myöhemmin Haavard ja Eirik perustivat yrityksen päästäkseen tuomaan tuotettaan markkinoille ja rakentaakseen maailman parhaan käyttöliittymien kehitysympäristön. Kirjain Q valittiin luokan etuliitteeksi, koska se näytti kauniilta Haavardin käyttämän tekstinmuokkaimen fontissa. T-kirjaimella viitataan englannin kielen sanaan toolkit, ohjelmoinnin työkalusarja. [6.]

Qt:sta julkaistiin saman ohjelmointirajapinnan sisältävä versio Windowsille ja Linuxille, ja se oli saatavilla alusta alkaen kahdella eri lisenssillä. Kaupallisten sovellusten kehitykseen tarvitaan kaupallinen lisenssi, mutta avoimen lähdekoodin sovelluksia varten ohjelmasta löytyy ilmainen versio. Vuosien kuluessa kehitysympäristö päivittyi ja monipuolistui huipentuen vuonna 2001 julkaistuun Qt 3.0:aan, jonka myötä ympäristö laajeni koskettamaan myös Mac OS X -käyttöjärjestelmää ja sulautetuissa laitteissa käytettäviä Linux-jakeluita. [6.]

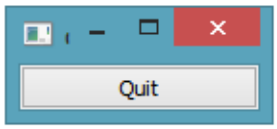
Nokia osti alkuperäisen Qt:ta kehittäneen yrityksen, norjalaisen Trolltechin, vuonna 2008 ja vaihtoi sen nimeksi Qt Development Frameworks. Siitä lähtien Nokia keskittyi tekemään Qt:sta pääasiallisen kehitysalustan laitteilleen. Nykyisellään Qt:n omistusoikeudet ovat Digialla, joka osti lisensointi- ja liikepalvelutoiminnan Nokialta vuonna 2011. Elokuussa 2012 julkistettiin kauppa, jonka mukaan Digia osti loputkin Qt-kehitysympäristöstä. [5.]

4.2 Qt:lla ohjelmointi

Qt-ohjelmoinnissa käytetään apuna koodaamista nopeuttavia valmiiksi olemassa olevia Qt:n omia luokkia. Luokkiin sisältyy muun muassa widgetejä, jotka käytännössä ovat graafisen suunnittelun puolella käytettäviä objekteja. Kuvassa 4 esitetään vasemmalla puolella lyhyt koodiesimerkki ja oikealla nähdään, mitä koodi saa aikaan suoritettaessa. Esimerkissä käytetään QPushButton-widgetiä, jolla luodaan painike. Quit-painikkeelle saadaan toiminnallisuus yhdistämällä painikkeelta tuleva signaali ”clicked()” valmiiseen slottiin eli toimintoon, joka tässä tapauksessa on ”quit()”.

```
1  #include <QApplication>
2  #include <QPushButton>

3  int main(int argc, char *argv[])
4  {
5      QApplication app(argc, argv);
6      QPushButton *button = new QPushButton("Quit");
7      QObject::connect(button, SIGNAL(clicked()),
8                       &app, SLOT(quit()));
9      button->show();
10     return app.exec();
11 }
```



Kuva 4. Qt:lla toteutettu Quit-sovellus

Qt Creatorilla työskenneltäessä graafinen suunnittelu on mahdollista ja helpottaakin huomattavasti käyttöliittymän ulkonäön ja käytettävyyden suunnittelua. Ulkonäön lisäksi Creatorin Designer-työkalulla voidaan yhdistää widgeteiltä tulevia signaaleja olemassa oleviin perustoimintoihin graafisesti. Kaikkea ei siis ole välttämätöntä tehdä koodia kirjoittamalla.

Vaikka perinteisestä C++:sta tutut elementit toimivat pohjana, ohjelmointi Qt-kehitysympäristössä eroaa suurimmalta osin omalla luokkakirjastollaan. Ohjelmointi kannat-

taakin usein aloittaa selaamalla Internetistä Qt Projectin kotisivuilta löytyviä dokumentaatioita, joista löytyy luokan perustietojen lisäksi koodiesimerkkejä.

Alkuun Qt vaikuttaa helposti sekavalta, varsinkin kun kirjain Q näyttää seikkailevan joka paikassa. Creatorissa on kuitenkin useita ohjelmoijaa avustavia ja työntekoa helpottavia elementtejä. Skooppien alku ja loppu hahmottuvat selkeästi, kun skoopin sulkevaa aaltosulkua napauttaa hiirellä. Selkeä, hillitty värikoodaus tekee koodista miellyttävämpää ja helpompaa luettavaa. Esimerkiksi muuttujien tietotyyppien fontin väri on keltainen, luokat erottuvat violetilla värillä ja merkkijonot sekä kommentit näkyvät vihreänä. Värien käytön lisäksi ympäristö auttaa ohjelmointia ennustavalla tekstinsyötöllä. Ohjelmoijan kirjoittaessa vasemman sulkeen ilmestyy oikeanpuoleinen sulje itsestään.

Virhetilanteilta ei ohjelmoitaessa voi välttyä, mutta onneksi se on otettu Qt:ssa huomioon. Kirjastosta löytyy QDebug-luokka, jolla ohjelmoija saa helposti lisättyä omia tarkistuksia sovelluksen ajoon. Luokasta löytyvällä qDebug-makrolla voidaan tulostaa muuttujien arvoja ajon aikana tai metodien palauttamia virheilmoituksia, joista on helpompi tulkita, mistä vika aiheutuu ja kuinka se voidaan välttää.

5 TIETOKANTASUUNNITTELU

Ennen varsinaisen suunnittelun aloittamista täytyi tietenkin perehtyä aiheeseen ja selvittää tietokantojen suunnitteluun liittyvää teoriaa, jotta pystyy välttämään aloittelijan kohtaamat ongelmat mahdollisimman hyvin ja välttyy virheiltä, tai ainakin sallii paremman todennäköisyyden huomata sellaisten synnyn. Vaikka tietokantoja voi suunnitella ja ideoida hyvin pitkälle lyijykynällä ja paperilla, ei toteutus kuitenkaan onnistu ilman muita työkaluja. Alkuun muutamana vaihtoehtona olivat Microsoft Office -ohjelmistopaketteihin kuuluvat Excel-taulukkolaskentaohjelma ja Access-tietokantaohjelma. Asiakkaan eräänä huomioonotettavana vaatimuksena lopullisessa tuotteessa oli helppokäyttöisyys, joka käytännössä heti rajasi vaihtoehdoista pois Excelin. Tietokantojen käsittelyyn tarkoitettu Access oli alkuun mahdollinen toteutusympäristö, mutta se karsiutui, kun tietokannan suorituskykyä arvioitiin. Ajan kanssa kasvavan talletettavan tiedon määrä alkaisi hidastaa tietokannan käsittelyä, jolloin ohjelma saattaa lakata toimimasta. Tämän lisäksi Office-paketti on maksullinen.

5.1 Työkalujen kartoitus

Nykytrendejä mukaillen lähdettiin kartoittamaan työkaluja avoimesta lähdekoodista löytyvillä ratkaisuilla. Käyttötarkoitukseen soveltuvia ehdokkaita löytyi lyhyellä selvittelyllä useita: MySQL, PostgreSQL ja SQLite. Tarkemman tietokantaohjelmiin tutustumisen jälkeen kattavimmat työkalut ja ominaisuudet löytyivät MySQL:stä, luottamusta herättävien referenssien lisäksi. Hyviin ominaisuuksiin kuuluvat muun muassa laaja yhteensopivuus eri ohjelmointikielien kanssa sekä graafiset työkalut tietokannan suunnittelua ja palvelimen hallinnointia varten. Näiden lisäksi MySQL on maailmanlaajuisesti runsaasti käytetty, jolloin tuotetukea ja apua ongelmatilanteiden ratkaisuun löytyy Internetistä suurella todennäköisyydellä monipuolisesti.

5.2 Käsiteanalyysi

Tietokannan suunnittelu alkoi käsiteanalyysin teolla. Analyysi pohjautui työn tilaajalta saatuihin säätöarvolomakkeisiin (engl. setup sheet), joista tietoja poimimalla mietittiin erilaisia kä-

sitteitä tietokannassa käytettäväksi. Kuvassa 5 on esitettynä alustavan käsiteanalyysin tuloksena syntyneet käsitteet, joista tietokannan taulut myöhemmin muodostuvat.

Kuva 5. Käsiteanalyysissä syntyneet käsitteet

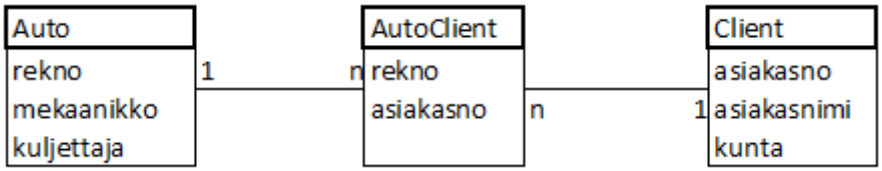
Kun tärkeimmät käsitteet alkavat olla koottuna, alkaa käsitteiden välisten suhteiden muodostaminen. Taulujen välille syntyy siten jonkinlainen suhde, jota tarvitaan, jotta tietokannan tietojen muokkaaminen onnistuu mahdollisimman mutkattomasti. Käsiteanalyysiä tehtäessä ja tauluja suunnitellessa on tärkeää pitää mielessä, että jokainen tieto tallennetaan vain kerran ja ainoastaan yhteen paikkaan tietokannan sujuvan käytettävyyden ja ylläpidon takaamiseksi. Toistoa on siis vältettävä, sillä saman tiedon muuttaminen kahdesta eri taulusta tuottaa ylimääräistä työtä tietokannan käsittelyssä. Tauluun tulevien sarakkeiden on tietenkin loogisesti ajateltuna liityttävä käsitteeseen, jota taulu edustaa.

5.3 Taulujen väliset yhteydet

Suurin osa suhteista oli aluksi moni-moneen-tyyppisiä yhteyksiä. Esimerkiksi kuvassa 6 Auto- ja Client-taulun välillä on moni-moneen-yhteys, koska yhdellä autolla voi olla monta omistaja-asiakasta. Toisaalta taas yhdellä omistaja-asiakkaalla voi olla monta autoa.

Kuva 6. Auto- ja Client-taulujen välinen yhteys

Moni-moneen-yhteys puretaan luomalla suhde taulujen välille välitaulun kautta. Kuvassa 7 näkyy purettu moni-moneen-yhteys, jolloin Auto- ja Client-taulujen väliin tehdään välitaulu AutoClient. Uuden taulun sarakkeiksi otetaan kopiot taulujen pääavaimista, muodostaen uuteen tauluun moniosaisen perusavaimen.



Kuva 7. Auto- ja Client-taulujen suhde välitaulun kautta

6 SETUP-TIETOKANNAN TOTEUTUS

Toteutuksen alkuvaiheessa käytiin läpi vaihtoehtoja tietokannan ja käyttöliittymän tekoon käytettävistä työkaluista. Suunnittelua ohjaamaan ja kokonaisuutta rajaamaan määriteltiin vaatimuksia ohjelmiston toiminnalle.

6.1 Työkalut

Insinöörityössä käytettävistä työkaluista ei ollut toiveita asiakkaan puolelta, jolloin päätös niistä jäi tekijän tehtäväksi. Mahdollisten vaihtoehtojen kartoituksen ja selvitystyön jälkeen työssä päätettiin käyttää MySQL Workbench 5.2:ta ja Qt 5.0.1:tä. Näihin työkaluihin johtaneet valintaperusteet on selostettu tarkemmin myöhemmin tässä dokumentissa.

MySQL Workbench 5.2

MySQL Workbench on visuaalinen työkalu tietokantojen suunnittelijoille, kehittäjille ja ylläpitäjille. MySQL Workbench yhdistää ja tarjoaa mahdollisuuden tietokantamallinnukseen, SQL-kyselyiden kehittämiseen sekä kattavat hallintatyökalut muun muassa palvelimen ylläpitoon, konfigurointiin ja käyttäjähallintaan. Ilmainen ohjelmakokonaisuus on saatavilla Windows-, Linux- ja Mac OS -käyttöjärjestelmille. [7.]

Suunnittelu

Workbenchillä tietokantojen suunnittelu, mallinnus, luominen ja hallinnointi tehdään visuaalisesti. Siihen sisältyy kaikki, mitä tiedon mallintaja tarvitsee luodakseen monimutkaisia ER-kaavioita, mallinnuksia ja takaisinmallinnuksia, ja lisäksi tärkeimmät ominaisuudet vaikean muutoshallinnan suorittamiseen ja dokumentointiin, jotka yleensä vaativat paljon aikaa ja vaivaa. [7.]

Kehitys

Ohjelma tarjoaa visuaaliset työkalut SQL-kyselyiden luomiseen, suorittamiseen ja tehostamiseen. SQL Editorilla voidaan selata yksittäisten taulujen sisältämiä tietoja, suorittaa ja testata SQL-kyselyitä tai hallinnoida tietokantayhteyksiä. [7.]

Qt 5.0.1

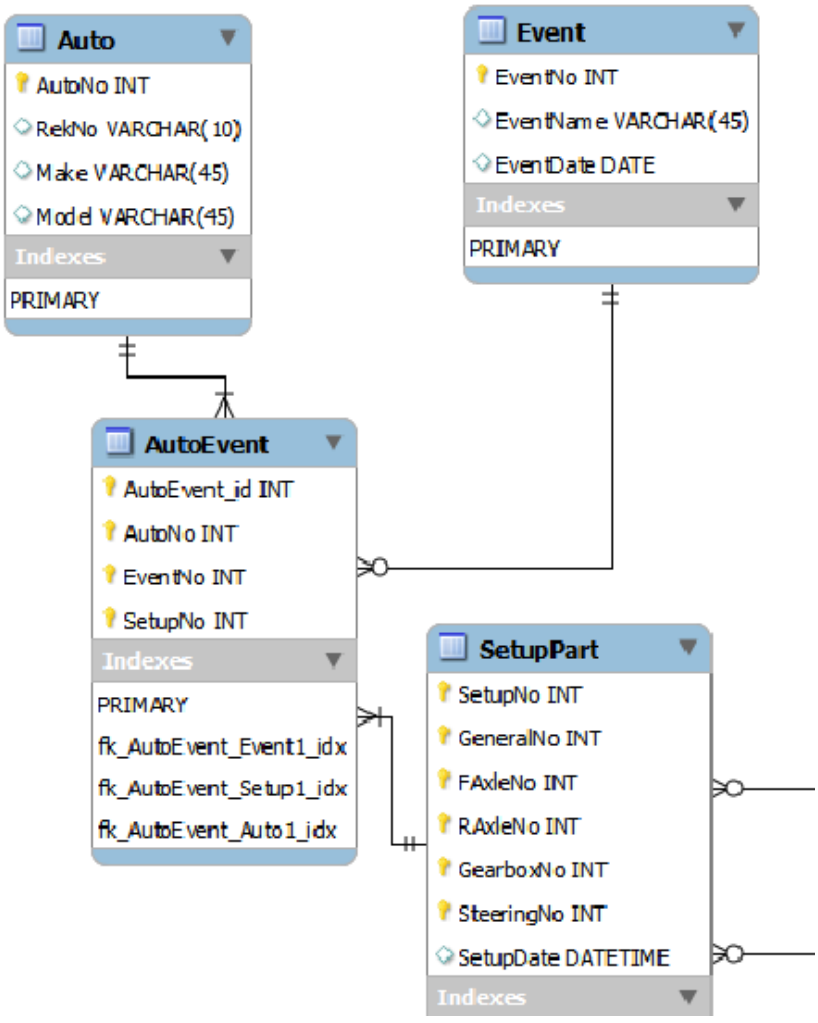
Qt on C++-kieleen perustuva sovelluksien ja erityisesti käyttöliittymien toteutukseen tarkoitettu kehitysympäristö, eli se ei itsessään ole ohjelmointikieli. Yli 450000 ohjelmoijaa yli 70 teollisuudenalalla ovat valinneet Qt:n kehittyneiden sovellusten ja laitteiden rakennukseen. [8.]

6.2 Vaatimusmäärittely

Asiakas esitti tarpeen järjestelmälle, jolla saadaan tallennettua digitaalisesti ralleissa käytetyt setupit autokohtaisesti. Näin ollen tiedot säilyvät pitkälle tulevaisuuteen ja ne ovat tarvittaessa helposti jälkikäteen löydettävissä. Tarpeeseen sopivimmalta ratkaisulta vaikuttaa tietokanta, joka räätälöidään tehtävään sopivaksi. Tietokanta ei kuitenkaan itsessään, riitä vaan sen käyttämiseen tarvitaan käyttöliittymä. Käyttöliittymän tärkeimpänä asiakkaan esittämänä vaatimuksena esiin nousi helppokäyttöisyys, jotta jokainen ohjelmaa tarvitseva oppii sitä käyttämään. Käyttöliittymän kautta pystytään lisäämään ja selailemaan setupeja sekä poistamaan niitä tietokannasta.

6.3 Tietokannan mallinnus

Kuvassa 8 nähdään tietokantakaavio, joka on koostettu aiemmin suunnitellun täydennetyn käsitemallin tuloksena syntyneistä käsitteistä. Myös aiemmin suunnitellut käsitteiden eli taulujen väliset yhteydet saadaan fyysisesti toteutettua. Yhteydet havainnollistuvat MySQL Workbenchissä taulusta tauluun vedetyillä ”harakanvarpailla”. Tietokannan mallinnustyökalu helpottaa huomattavasti tietokannan rakenteen hahmottamisessa ja vältetään ongelmia luomisvaiheessa.



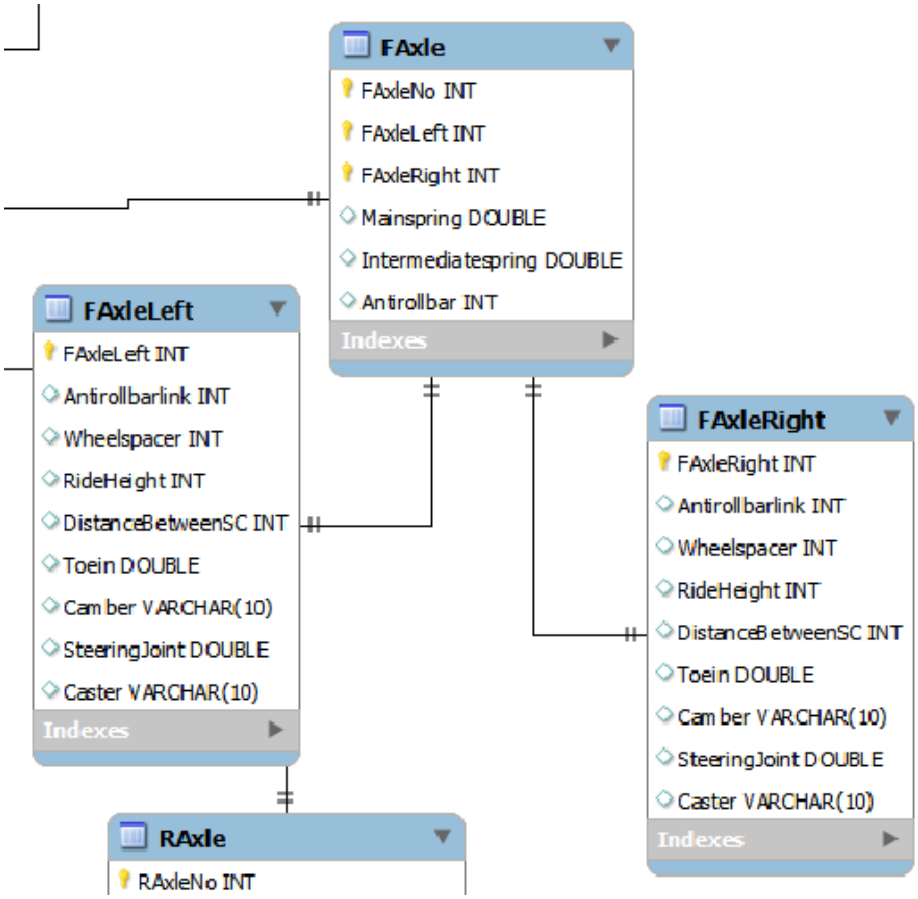
Kuva 8. MySQL Workbenchillä toteutettu ER-tietokantakaavio

Taulujen perusavaimet esitetään kaaviossa keltaisella avainkuviolla. Muut tauluihin tallennettavat tiedot erottuvat kaaviosta vinoneliöllä.

Workbench toimii hyvänä apuna yhteyksiä toteutettaessa, eikä se anna luoda yhteyttä, jos taulujen välinen logiikka on pielessä. Yhteyksiä suunniteltaessa on otettava huomioon usein toistuvat tiedot, joita ei siten tarvitse tallettaa kuin kerran. Tällaisia tietoja setup-tietokannassa ovat auton (Auto-taulu) ja tapahtuman (Event-taulu) tiedot. Auton ja tapahtuman välillä on moni-moneen-yhteys, koska yksi auto voi liittyä moneen tapahtumaan, ja yksi tapahtuma voi liittyä moneen autoon. Niitä yhdistämään perustetaan välitaulu AutoEvent.

AutoEvent-tauluun liittyy SetupPart-taulu, joka kokoaa yhteen setupiin liittyvät tiedot. Käytännössä tämä tapahtuu niin, että SetupPart-tauluun tallennetaan perusavaimet alemman tason tietotauluista. Yhdellä autolla voi yhdessä tapahtumassa olla useammat eri setup-arvot, jolloin ne erotellaan toisistaan kellonajan perusteella.

Tietotaulut koostuvat pääasiassa sarakkeista, joihin talletetaan käyttäjän syöttämää tietoa. Kuvassa 9 näkee etuakselin säätöarvojen talletukseen suunnitellun tietotaulun. Koska samoja arvoja kerätään sekä vasemmalta että oikealta puolelta autoa, on kummallakin puoliskolla oma taulu, jotka yhdistetään välitaululla. Sarakkeen otsikon jälkeen lukee tietotyyppi, eli minkä tyyppistä tietoa sarake ottaa vastaan. FAxle-taulun Mainspring-sarake hyväksyy double-tyyppisen arvon - desimaalinumeron.



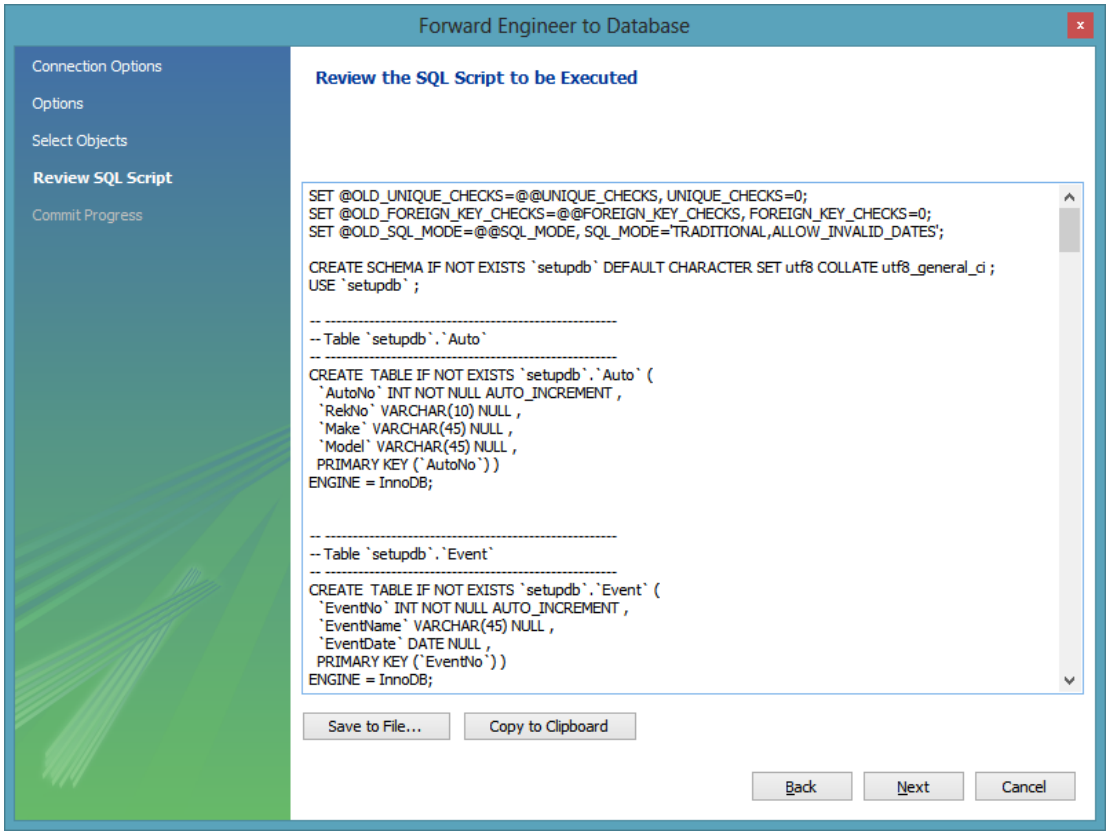
Kuva 9. Tietotaulun kuvaus ER-tietokantakaaviossa

6.4 Tietokantakaavion kääntäminen - tietokannan luominen

Valmiista tietokantakaaviosta saadaan käännettyä toimiva tietokanta Workbenchistä löytyvällä työkalulla. Toimenpiteen nimi on ”Forward Engineer”, joka käytännössä muuntaa piirretyn tietokantamallin MySQL-palvelimelle toimivaksi tietokannaksi.

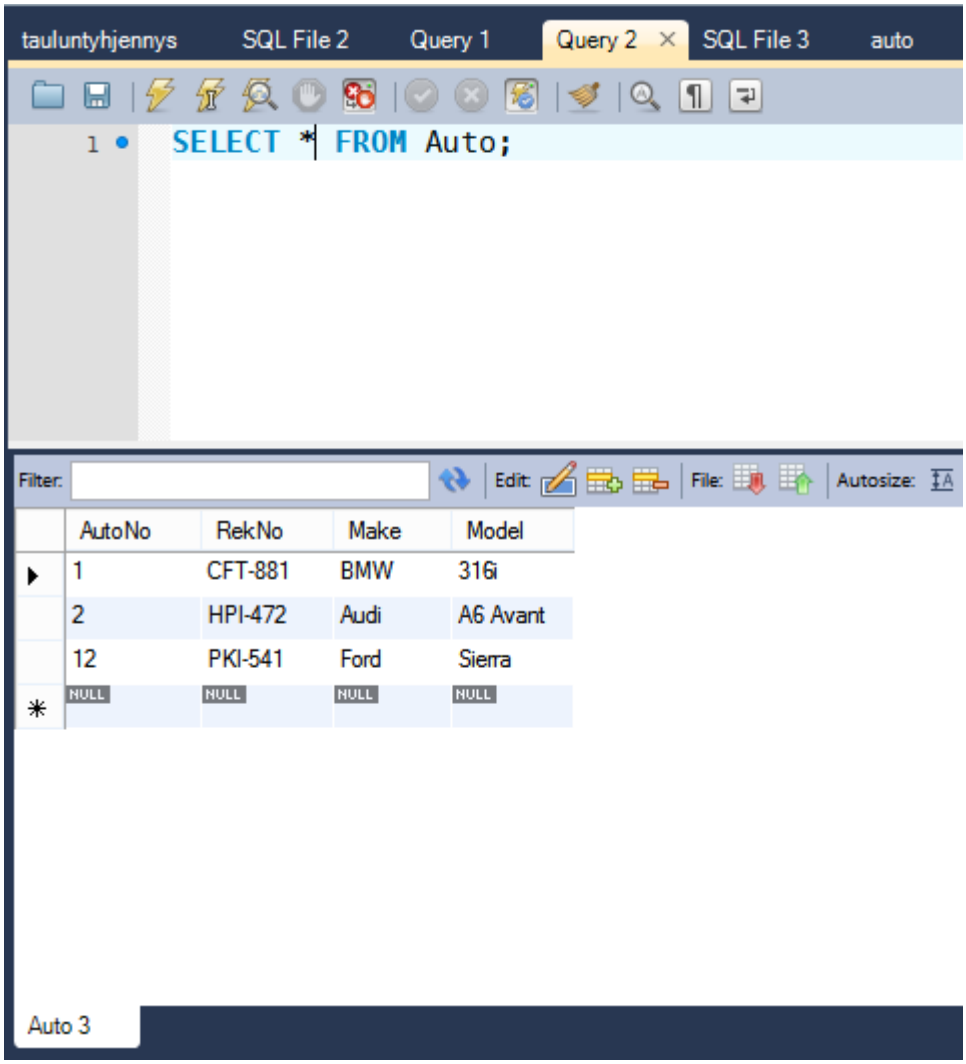
Tietokanta luodaan SQL-kyselyitä käyttäen ja onnistuu myös MySQL:n mukana tulevan komentorivityökalun avulla. ”Forward Engineer” luo SQL-kyselyt käyttäjän puolesta ja suorittaa ne, jolloin palvelimelle syntyy tietokanta. Työkalu helpottaa tietokannan luomista ja poistaa tarpeen SQL-kyselyiden kirjoitukseen, jolloin kirjoitusvirheiden aiheuttamat ongelmat poistuvat. Kääntäessä työkalu huomaa, jos yhteyksien logiikassa on virheitä ja antaa niistä virheilmoituksen lopettaen kyselyiden suorittamisen. Virheilmoituksen avulla käyttäjä voi jäljittää vian syyn ja tehdä korjauksen.

Kuvassa 10 näkyy ”Forward Engineer” -työkalun välilehti ennen SQL-kyselyiden suoritusta. Käyttäjä näkee ajettavat komennot ja voi tarvittaessa tehdä niihin muutoksia ennen suoritusta.



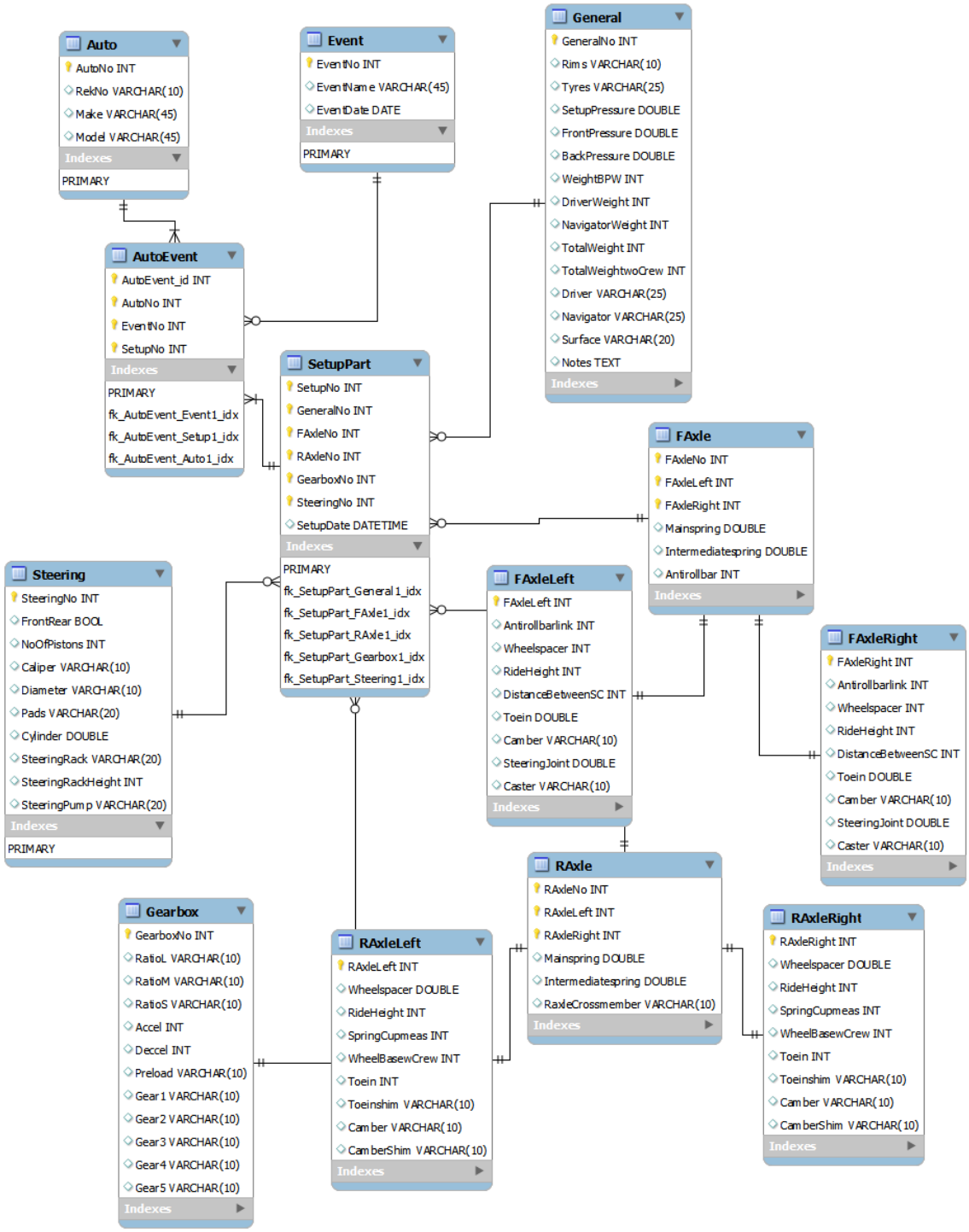
Kuva 10. "Forward Engineer" -työkalun ikkuna

SQL Editorilla pääsee käsiksi palvelimella olevaan tietokantaan. Editor mahdollistaa SQL-kyselyiden suunnittelun ja testaamisen. Tietokannan taulujen sisältämiä tietoja voi selata ja muokata. Kuvassa 11 on SQL Editorin näkymä SQL-kyselyn suorittamisen jälkeen. ”SELECT * FROM Auto;” -kyselylauseke palauttaa suoritettaessa kaikki sarakkeet ja rivit Auto-taulusta.



Kuva 11. SQL Editorin näkymä kyselyn suorituksen jälkeen

Viimeistelty relaatiotietokannan tietokantakaavio sisältää 13 taulua, joista jokainen on jotain kautta toisiinsa yhteydessä. Taulut ja niiden väliset yhteydet havainnollistuvat parhaiten kuvasta 12. Setupeissa käytettävät tiedot on jaettu osa-alueisiin (”General”, ”Steering”, jne.), jotka tietokannassa erotellaan selkeyden ja käsittelyn helpottamisen vuoksi omiin tauluihinsa.



Kuva 12. Viimeistelty ER-tietokantakaavio

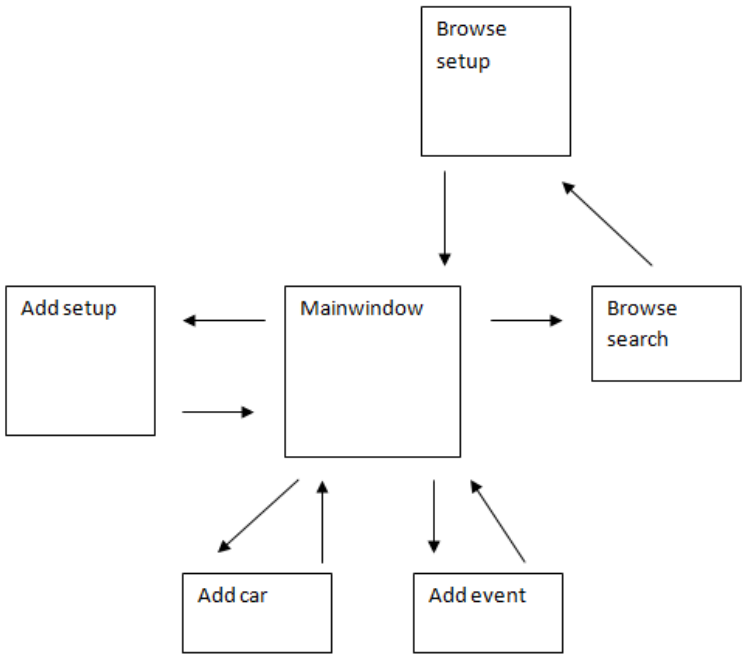
6.5 Käyttöliittymän ohjelmointi

Tietokantayhteys

Ennen ohjelman toiminnallisuuksien toteutusta piti käyttöliittymän ja tietokannan välinen yhteys saada toimimaan. Qt:ssa ei ole vakiona MySQL-tietokantayhteyden vaatimia ajureita, mutta valmiudet ja tuki ovat olemassa. Qt on onneksi suosittu ympäristö kehittäjien joukossa ja Internetistä löytyy paljon dokumentaatiota kaikenlaisiin ongelmatilanteisiin. Päättäväisen tiedonhaun tuloksena löytyivät yksityiskohtaiset ohjeet, joiden avulla ajurien kääntäminen onnistui.

Dialogit

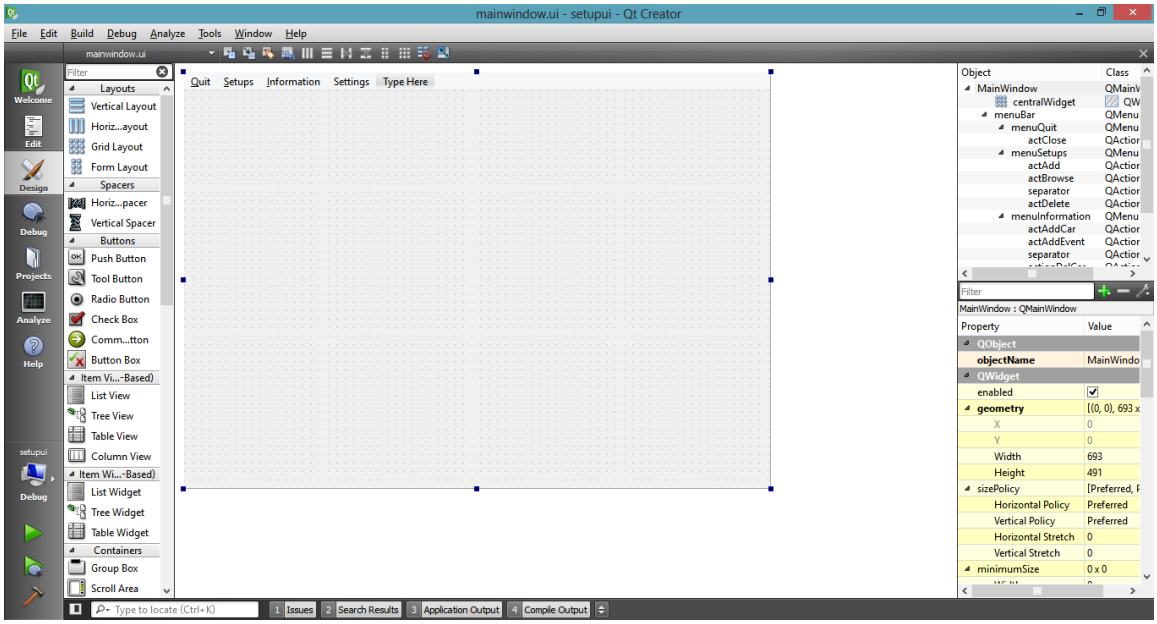
Käyttöliittymän dialogien välisiä siirtymiä ja toimintaa auttaa hahmottamaan suunnittelun aikana tehty käyttöliittymäkartta. Ohjelman käyttö aloitetaan kuvan 13 keskellä sijaitsevasta aloitusikkunasta (Mainwindow).



Kuva 13. Käyttöliittymäkartta

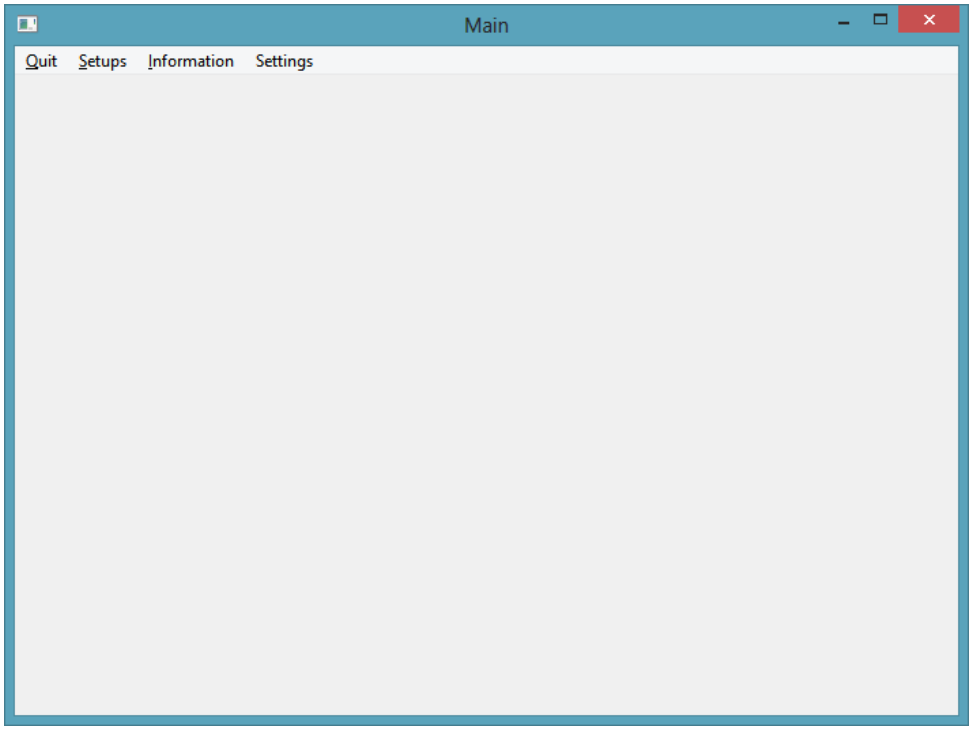
Qt:lla toteutettu graafinen käyttöliittymä koostuu dialogeista. Dialogit on helpointa suunnitella graafisesti Creatorista löytyvällä Designer-työkalulla. Ohjelman toiminnallisuuksien mu-

kaan dialogeihin lisätään widgetejä, joita ovat muun muassa painikkeet, teksti- ja numerokentät. Kuvassa 14 näkyy Designer-työkalu, jossa auki olevaa dialogia voi muokata. Vasemmalla puolella olevasta listasta saa hiirellä vedettyä ja pudotettua widgetejä dialogiin.



Kuva 14. Qt Creatorin Designer-työkalu

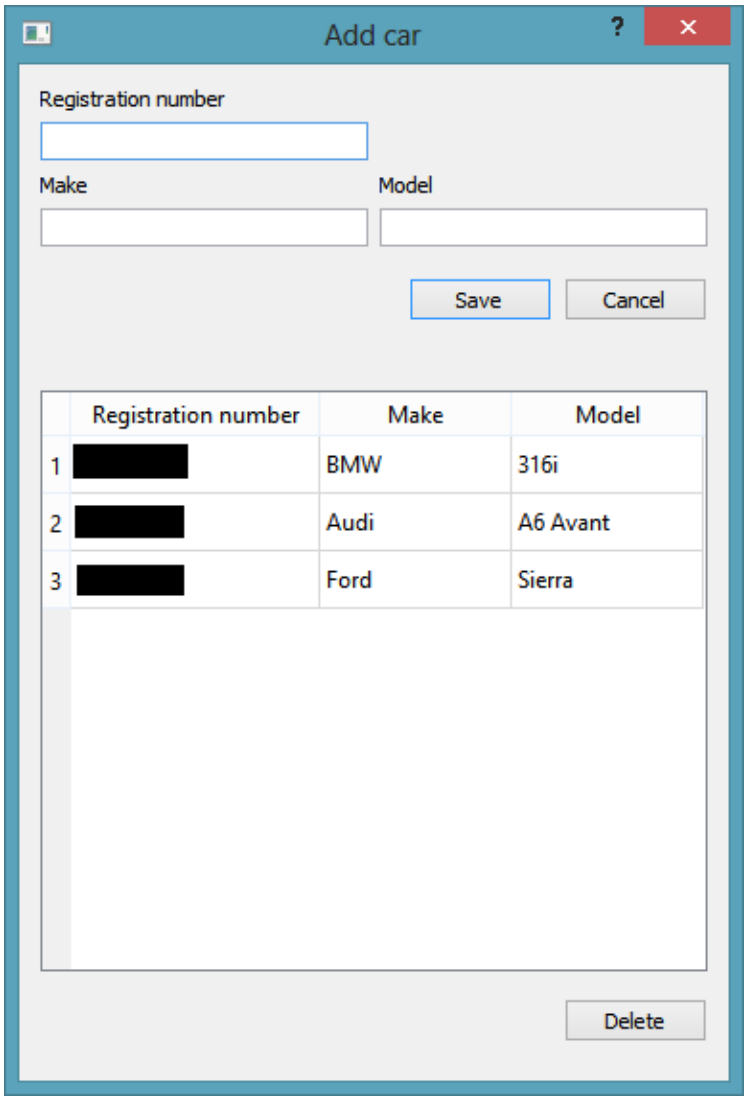
Setup-käyttöliittymään tehty aloitusikkuna on hyvin yksinkertainen, kuten kuvasta 15 näkee. Tämä ikkuna aukeaa ensimmäisenä, kun ohjelma avataan. Ainoat käyttäjälle näkyvät toiminnallisuudet löytyvät ylärivin valikoista. ”Quit”-valikosta ohjelma suljetaan. ”Setups”-valikon kautta avataan lomake, jolla setup-tietoja lisätään tietokantaan tai valinnaisesti selataan aiemmin syötettyjä tietoja. ”Information”-valikosta lisätään usein toistuvia perustietoja, kuten tarvittavat autot ja tapahtumat. ”Settings”-valikosta päästään vaikuttamaan tietokantayhteyden asetuksiin.



Kuva 15. Setup-käyttöliittymän aloitusikkuna

Taustalla, käyttäjän näkymättömissä, luodaan valmiiksi yhteys tietokantaan, mutta sitä ei vielä avata. Yhteyden avaus tapahtuu ainoastaan silloin, kun siihen on tarve ja yhteys suljetaan heti käytön jälkeen tietoturvasyistä.

Ennen kuin setup-tietoja pääsee syöttämään tietokantaan, lisätään ohjelmaan auton tiedot. Auton lisäys tapahtuu ”Add car” -ikkunassa, jonka saa auki aloitusikkunan ”Information”-valikosta. Autot yksilöidään rekisterinumeron perusteella, ja sen lisäksi voidaan tallettaa auton merkki ja malli, kuten kuvasta 16 nähdään. Samasta ikkunasta onnistuu myös auton tietojen poisto, jolloin tietokannasta hävitetään kaikki autoon liittyvät tiedot.

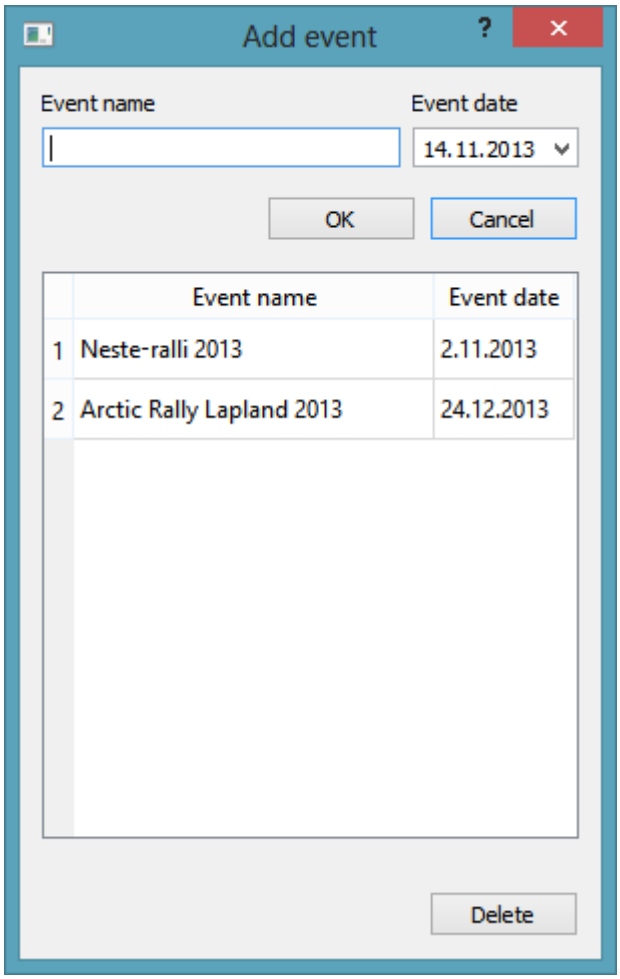


Kuva 16. Add car -ikkuna

Ikkunan avautuessa avataan yhteys tietokantaan, josta haetaan tiedot taulukkonäkymään aiemmin tallennetuista autoista. Haun jälkeen tietokantayhteys suljetaan. ”Save”-painiketta painettaessa uuden auton tiedot tallennetaan tietokantaan ja taulukkonäkymä päivitetään. Poistettaessa auton tietoja tulee poistettava auto valita ensin painamalla poistettavaa riviä ja sen jälkeen klikkaamalla ”Delete”-painiketta. Poistettavaa riviä voidaan painaa minkä sarakkeen kohdalta tahansa. Taulukkonäkymä päivitetään myös tietojen poiston jälkeen. Ohjelma

varmentaa käyttäjältä, haluaako hän todella poistaa tiedot, jolloin vältytään vahinkopainallusten aiheuttamalta tuholta. Taulukkonäkymästä ei voi muokata tietoja suoraan.

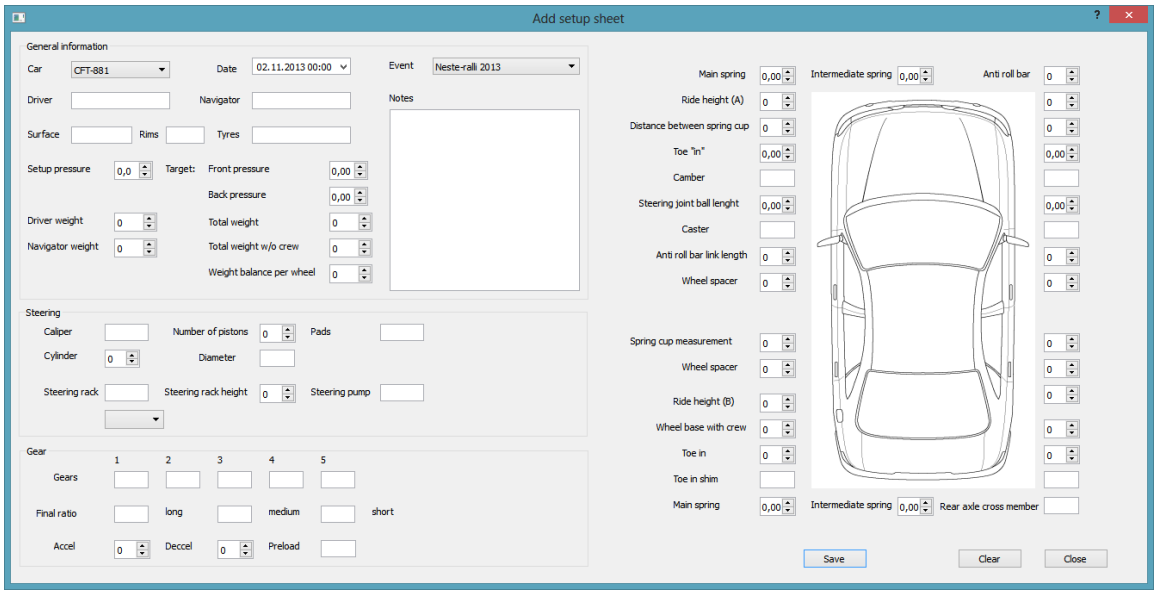
Auton lisäämisen jälkeen tarvitaan vielä tapahtuma, johon setup-tiedot yhdistetään. ”Information”-valikosta avataan ”Add event”-ikkuna, kuvassa 17, johon kirjoitetaan tapahtuman nimi ja asetetaan päivämäärä, jolloin tapahtuma on ollut. Tarvittaessa luotu tapahtuma voidaan myös poistaa.



Kuva 17. Add event -ikkuna

Tapahtumien käsittely toimii samaan tapaan kuin autojen tietojen käsittely.

Uusi setup lisätään avaamalla ”Setups”-valikosta ”Add setups” -ikkuna. Ikkunan avautuessa muodostetaan yhteys tietokantaan ja haetaan pudotusvalikkoihin kaikki tietokannasta löytyvät autot sekä tapahtumat. Näitä ei siten tarvitse joka kerta kirjoittaa uudelleen, vaan valmiina olevia tietoja voi käyttää pohjana uudelle setupille. ”Add setups” -ikkunassa tietojen syöttämistä selkeyttää kuva autosta, kuten kuvasta 18 näkee.



Kuva 18. Add setups -ikkuna

Tietojen syöttämisen jälkeen tallennus tapahtuu painamalla ”Save”-painiketta. Tällöin kenttiin syötetyt tiedot siirretään tietokantaan, jokainen omalle paikalleen. Ohjelmakoodissa siirto on toteutettu taustalla suoritettavilla SQL-kyselyillä, joille on luotu oma luokka. Tiedot on jaettu osa-alueisiin, joille jokaiselle löytyy tietokannasta oma tietotaulu.

SQL-kyselyt

Qt:lla ohjelmoitaessa käytetään ohjelmointikielenä C++:aa, joka mahdollistaa luokkien käytön ja olio-ohjelmoinnin. Luokkien käyttö helpottaa ohjelmointia sellaisessa tapauksessa, jossa samaa toiminnallisuutta tarvitaan useammassa eri yhteydessä. Samaa koodia ei siten tarvitse kirjoittaa moneen eri kohtaan, vaan sitä voidaan käyttää kutsumalla.

Käyttöliittymään luotiin oma luokka SQL-kyselyille, joita käytetään, kun tietokantaan tehdään muutoksia kuten tietojen lisäyksiä, päivityksiä tai poistoja. Luokasta löytyy jokaiselle taululle oma tallennus- ja päivitys-metodi. Painettaessa ”Save”-painiketta ”Add setups” -

ikkunassa käydään tallennusmetodit läpi, jolloin syötetyt tiedot siirretään talteen tietokantaan. Qt:ssa SQL-kyselyt voidaan toteuttaa kuvassa 19 näkyvällä tavalla.

```
void SqlQueries::saveEvent(QString eventName, QString eventDate)
{
    QSqlQuery bindQuery;
    bindQuery.prepare("INSERT INTO Event (EventName, EventDate)"
                      "VALUES (:EventName, :EventDate)");
    bindQuery.bindValue(":EventName", eventName);
    bindQuery.bindValue(":EventDate", eventDate);

    bool bTest = bindQuery.exec();

    if (false == bTest)
    {
        qDebug()<<bindQuery.lastError().text();
    }
}
```

Kuva 19. Tapahtuman tallentava saveEvent-metodi

Metodia kutsuttaessa sille tuodaan argumenteiksi käyttäjän ”Event”-ikkunaan syöttämät kenttien tiedot. Prepare-rivillä luodaan käytettävä SQL-kyselylauseke, jota kirjoitettaessa täytyy olla huolellinen, varsinkin jos lausekkeen väliin tehdään rivinvaihtoja. Lausekkeen sisällöstä on hankalaa paikallistaa virhettä muuten kuin oikolukemalla, sillä lausekkeen on oltava pilkulleen oikein. Prepare käsittelee lauseketta QString-tyyppisenä muuttujana. Bindvalue-riveillä tuodut attribuutit sidotaan käsiteltävän tietokannan taulun sarakkeisiin.

Kysely suoritetaan lopuksi exec-komennolla. Tallennusmetodeihin on varmuuden vuoksi lisätty tarkistus, jossa debuggaukseen käytetty qDebug-makro ilmoittaa kyselyn virheviestin, jos kysely palauttaa epätoden arvon, eli suorittaminen ei onnistu.

7 TYÖN TULOKSET

Työn teko oli opettava kokemus, varsinkin kun käytetyistä ohjelmista ei tekijällä ollut aiempaa kokemusta. Oli mielenkiintoista huomata, kuinka edistyksellisiä ja käyttökelpoisia avoimen lähdekoodin työkalut osaavat olla. Suunnittelun vaatima aika ja ajatustyön määrä yksin työskenneltäessä yllättivät.

Työn tuloksena saatiin suunniteltua ja toteutettua toimiva relaatiotietokanta, joka tehtiin käyttäen apuna teorioita tietokanta-ammattilaisten kirjoittamista teoksista. Työn teon ja teorian opettelun myötä ymmärrys tietokantoihin ja niiden suunnitteluun kasvoi. Tietokanta on monikäyttöinen työkalu, ja sen monipuoliset räätälöivissä olevat mahdollisuudet tekevät siitä varteenotettavan vaihtoehdon tietojen tallennuksen näkökulmasta. Tietokannan käyttökohteet ovat lukemattomat.

Käyttöliittymän toteuttaminen kehitti tekijän alkeellisella tasolla olleita ohjelmointitaitoja. Aiemman käyttöliittymien suunnittelun kokemuksen puuttuminen ei juuri häirinnyt graafista ulkoasua toteutettaessa. Nykypäivän Windows-käyttöympäristön tietokoneohjelmat noudattavat pitkälle samoja kaavoja, joita soveltaen syntyi ohjelman aloitusikkuna. Käyttäjälle tutun mallin toisintaminen oli ohjelman helppokäyttöisyyden ja nopean käytön oppimisen takia luonnollista. Käyttöliittymän toiminnallisuuksien ohjelmointi oli mielenkiintoista ja korosti sitä totuutta, ettei ohjelmointia opi muuten kuin ohjelmoimalla.

Suurimpana yksittäisenä ongelmana oli tietokantayhteyden saaminen toimintaan käyttöliittymän ja tietokannan välille. Tähän työvaiheeseen kului luvattoman paljon aikaa varsinkin jälkikäteen katsottuna, koska ohjeet ovat kuitenkin hyvin selkeät. Työn alkuvaiheessa tekijän kokemuspohja oli tietenkin huomattavasti pienempi. Tämän takia myös käyttöliittymän ohjelmoinnissa vastaan tulleet ongelmat hidastivat työn etenemistä. Seinän tullessa jossain kohti vastaan oli kuitenkin mahdollista keskittyä välillä tekemään jotain muuta työn osa-aluetta.

Käyttöliittymä ei työn loppuvaiheessa täyttänyt kaikkia sille asetettuja tehtäviä, mutta ohjelmoimalla lisää tarvittavat ominaisuudet saataisiin toimimaan. Jatkokehitykseen olisi siis aihetta, jotta ohjelma saataisiin käyttökuntoon. Testauksen kautta nousisi esiin todennäköisesti useita huomioon otettavia lisäyksiä ohjelman toimintaan.

8 YHTEENVETO

Työssä keskityttiin aluksi selventämään tietokantojen teoriaa, jotta suunnitteluvaiheessa ymmärretään, mitä ollaan tekemässä ja miksi. Teorian ymmärtäminen myös auttaa selkeyttämään, millaiseen lopputulokseen pyritään. Tietokannan rakenteen sisäistäminen ja kokonaisuuden kasaaminen pienistä paloista sekä opettaa, että helpottaa työn etenemistä. Huolellinen suunnittelu varmistaa jatkossa tietokannan luotettavan toiminnan ja mahdollistaa jälkikäteen tehtäviä muutoksia. Käyttöliittymän suunnittelu on erilainen prosessi, jossa keskitytään pääasiassa ulkoisiin tekijöihin, jotka ovat suoraan verrannollisia käytettävyyteen.

Yleiskäyttöistä tietokantaa ei ole olemassa, vaan sellainen on aina toteutettava käyttötarkoitukseen sopivaksi. Insinöörityössä syntynyt tietokanta on rakenteeltaan alkuperäisten vaatimusten mukaan käyttökelpoinen ja toimiva. Käyttöliittymä sisältää tärkeimmät toiminnallisuudet, ja sillä voidaan todeta tietokannan toimivuus käytössä.

Lopputyön teko sujui pääasiassa hyvin, vaikka projektin aikataulu venyi reilusti yli alkuperäisen suunnitelman. Uusien työkalujen käytön opettelu vaati oman aikansa, mutta työ tekijäänsä opettaa, niin kuin sanotaan. Vähäinen kokemus ohjelmoinnista aiheutti omat ongelmansa, eikä työ siltä osin ollut alkuun sujuvaa ennen kuin jotain oli saatu aikaiseksi. Aloittaminen on aina hankalinta, varsinkin kun ei ole tietoa, mistä kannattaa lähteä liikkeelle - kokemuksen tuoma suunnitelmallisuus puuttuu. Projekti on opettanut paljon varsinkin tiedonhausta ja sen tärkeydestä, sekä vahvistanut luottamusta yksilön kykyihin, joilla uusista haasteista pystytään suoriutumaan.

Tuloksena luotu tietokanta täyttää tehtävänsä, mutta käyttöliittymän puutteiden takia ohjelma ei kokonaisuutena ole käyttövalmis. Tietojen poistaminen ”keskeltä” tauluja ei nykyisellään onnistu, jolloin indeksinumeroiden kanssa tulee ongelma. Tämän tärkeän perusominaisuuden lisäksi ohjelma tarvitsee suunnitelmallista testausta, jolla lisätään toimintavarmuutta ja käytön aikana mahdollisesti esiintyviä vikatilanteita minimoidaan. Käyttäjän tekemien toimintojen varmentaminen vaatii myös lisäyksiä, jotta vältytään esimerkiksi tiedon tahattomalta häviämiseltä suljettaessa dialogeja.

LÄHTEET

1. Hovi Ari - Huotari Jouni - Lahdenmäki Tapio, 2005, Tietokantojen suunnittelu & indeksointi, ISBN: 978-951-846-262-3.

2. Hernandez Michael James, 2003, Database Design for Mere Mortals, ISBN: 0-201-75284-0.

3. Introduction to SQL, [WWW-dokumentti] <http://www.w3schools.com/sql/sql_intro.asp> (Luettu 28.2.2013)

4. Kalimo Anna, 1995, Graafisen käyttöliittymän suunnittelu, ISBN: 951-762-382-8.

5. Qt (framework), [WWW-dokumentti] <http://en.wikipedia.org/wiki/Qt_(framework)> (Luettu 13.3.2013)

6. Blanchette Jasmin - Summerfield Mark, 2008, C++ GUI Programming with Qt 4, ISBN: 0-13-714397-4.

7. MySQL Workbench, [WWW-dokumentti] <http://www.mysql.com/products/workbench/> (Luettu 11.4.2013)

8. Getting Started?, [WWW-dokumentti] <http://qt-project.org/resources/getting_started> (Luettu 16.4.2013)